(第1回 BitVisor Summit)

BitVisorによる動画像を対象とした著作権保護

小清水 滋，新城 靖，板野 肯三（筑波大学），
榮樂 英樹，松原 克弥（株式会社イーゲル）

背景と目的 (1/2)

現在、PCへの動画配信が普及して久しいが、依然として著作権保護の問題が残されている

代表的な著作権管理技術

- マイクロソフトのデジタル著作権管理技術
- デジオンのDiXiM
  - ハードディスクレコーダ等で使われるプロトコルを用いている

- ユーザプロセスの権限で実現されているためユーザは安心して利用できる
- リバースエンジニアリングで保護を破ることが可能

ユーザ寄りの著作権保護

背景と目的 (2/2)

- ソニーのPlayStation3
  - ユーザがインストールしたOSのプロセスからはDVDのデータがコピーできない
  - 著作権保護は十分機能している

- 独自の仮想計算機モニタ(VMM)を用いており保護は破れない
- VMMという高い権限で実現されおりユーザは安心して利用できない

権利者寄りの著作権保護

本研究

- 動画像のストリーミング再生中にユーザ、または、悪意の第三者に動画データをコピーされないことを保証
- ユーザが安心して利用できる著作権保護の実現

双方に中立な著作権保護

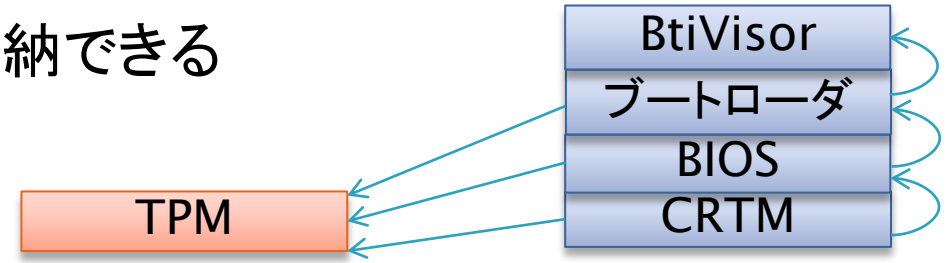
本研究のTrusted Computing Base

- ユーザPC
  - ソフトウェア：CRTM、BIOS、ブートローダ、BitVisor
  - ハードウェア：TPM
- 外部のサーバ
  - 検証サーバ、認証局、プライバシ認証局

- 前提条件
  - ハードウェアに対する攻撃は想定しない

Trusted Bootとは

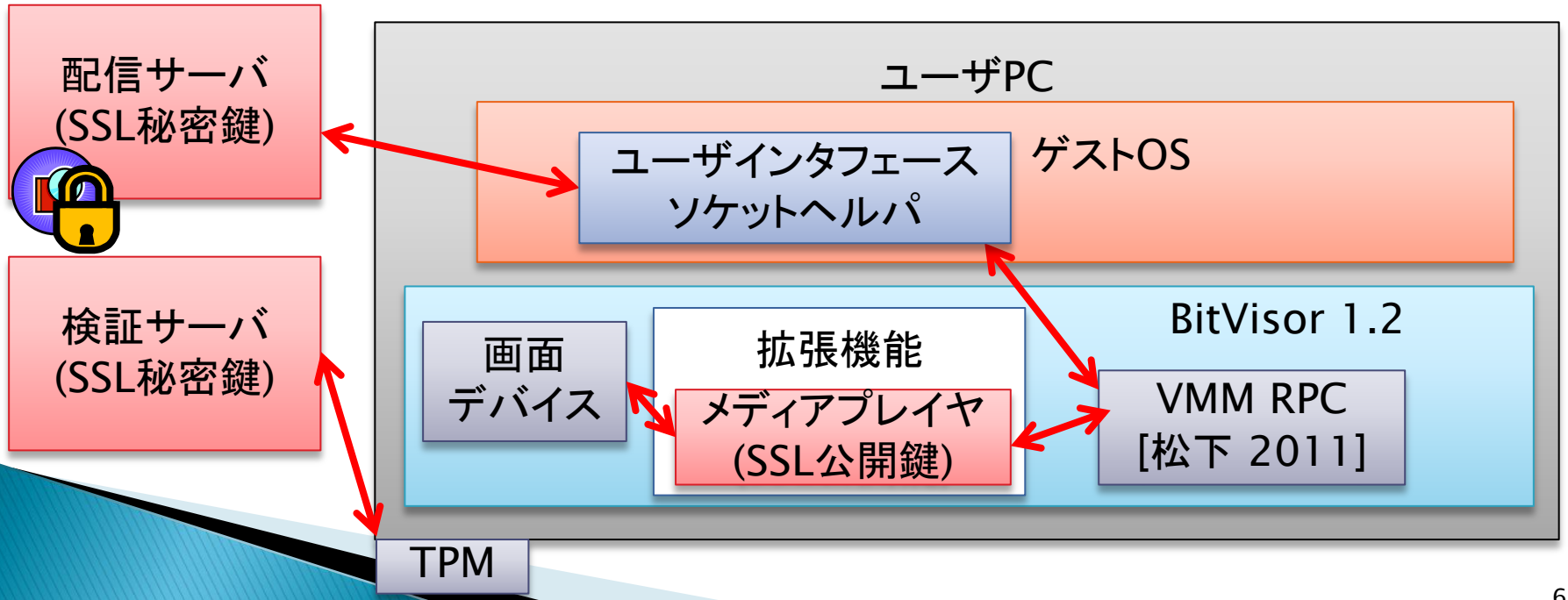
- ユーザ環境のソフトウェアの完全性を起動後に検証することが可能な起動方法
- TPMを用いることにより実現
- TPM (Trusted Platform Module)
  - セキュリティチップ
  - TPMの秘密鍵はハードウェア所有者ですら取り出し不可能
  - TPMの公開鍵は検証者が確認できるプライバシ認証局が適切に管理
  - PCRにハッシュ値を格納できる



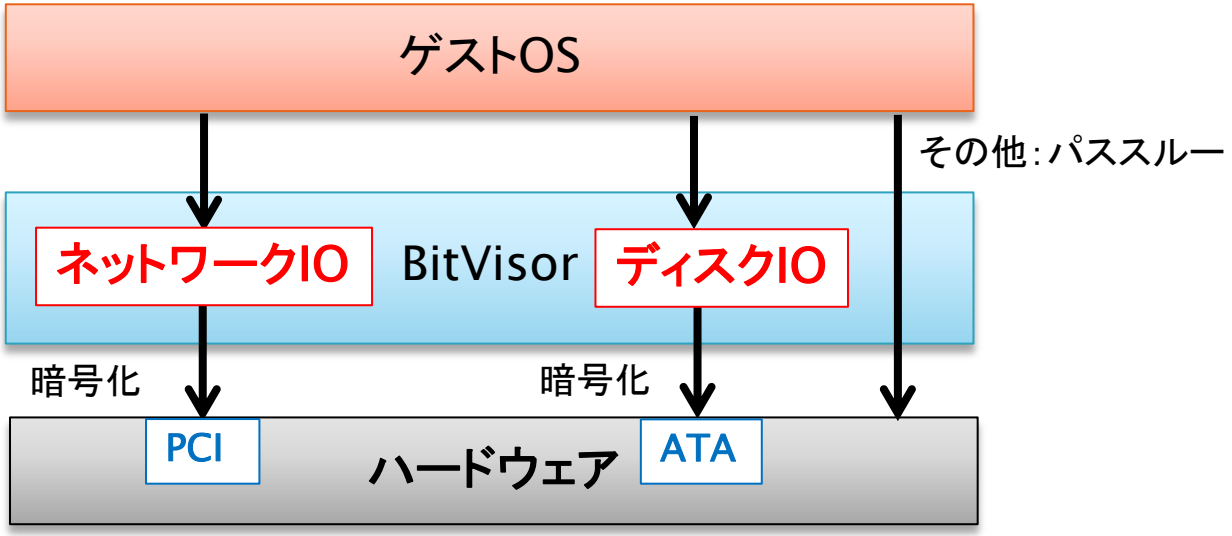
次に読み込むモジュールのハッシュ値をTPMに保存し、信頼の連鎖を作りながら起動

BitVisorによる著作権保護手法の設計

- ユーザ用、コンテンツプロバイダ（CP）用の2つの実行環境がある
  - VMMとゲストOSのメモリ空間は隔離されている
  - ユーザ用：ゲストOS　CP用：VMMの拡張機能
- メディアプレイヤと配信サーバはSSLを用い通信できる
- 動画の再生時に画面デバイスはメディアプレイヤで専有する
- VMMの完全性はユーザとCPの双方から検証できる(Trusted Boot)



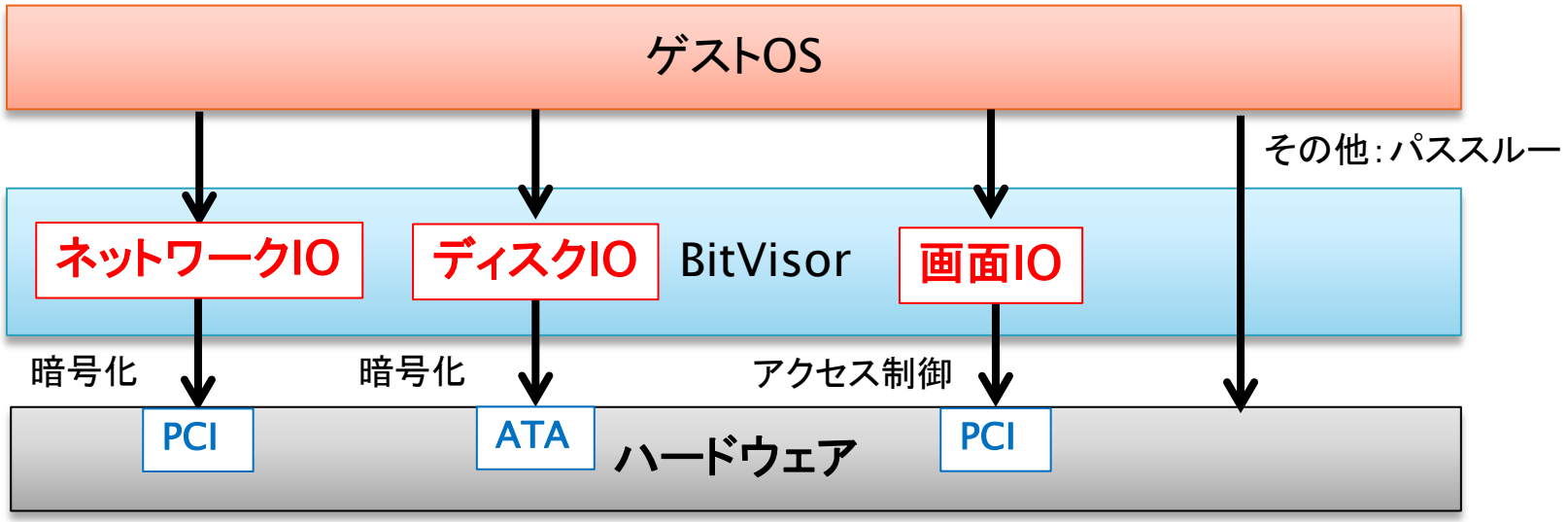
準パススルー型VMM BitVisor



ディスク、ネットワークIOを暗号化

BitVisorによる画面デバイスの専有

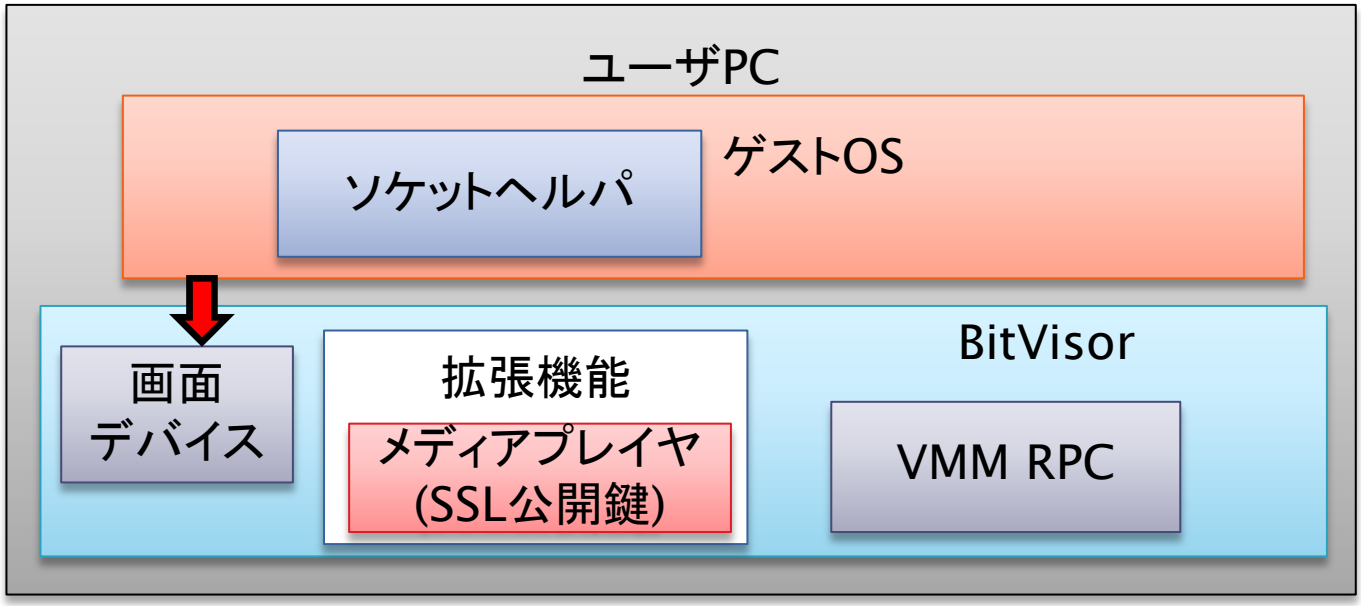
- 画面デバイス
  - グラフィックアクセラレータもPCIデバイスが一般的



脅威モデル（1/3）

- ゲストOSから画面デバイスへアクセスして画像データをコピーする攻撃
  - ゲストOSとBitVisorのメモリ空間は隔離されている

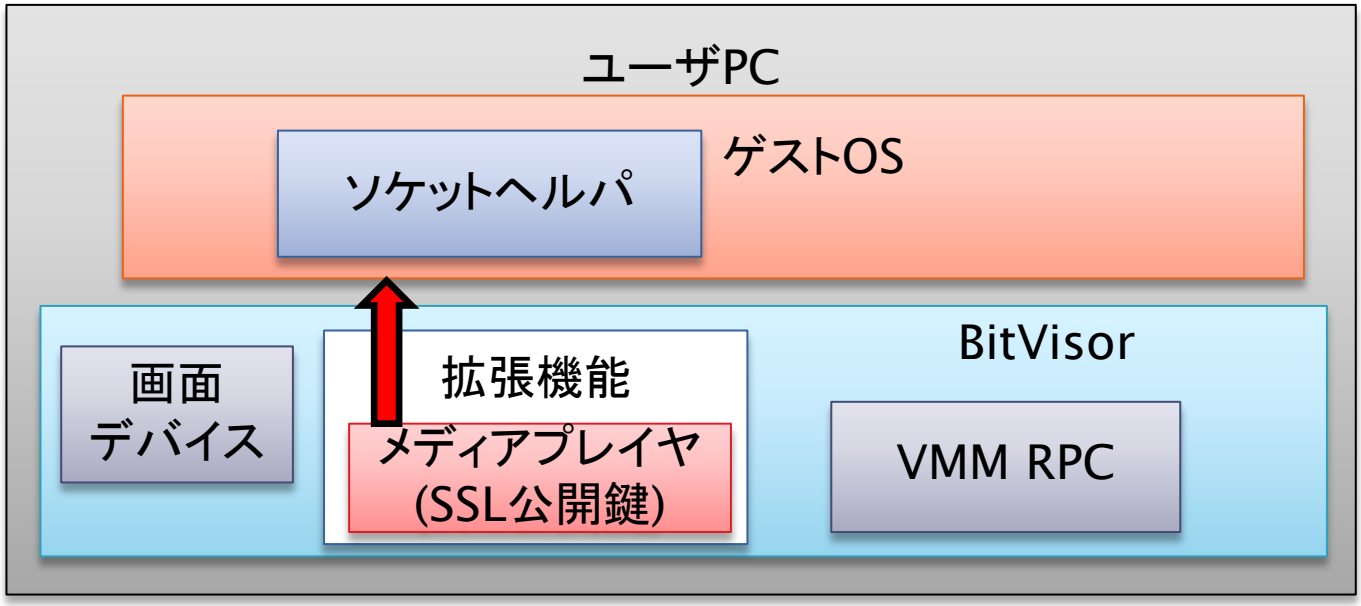
⇨ BitVisorがメディアプレイヤへ画面デバイスを専有させることで防ぐ



脅威モデル（2/3）

- メディアプレイヤがゲストOSのデータへアクセスする攻撃
  - ゲストOSとBitVisorのメモリ空間は隔離されている

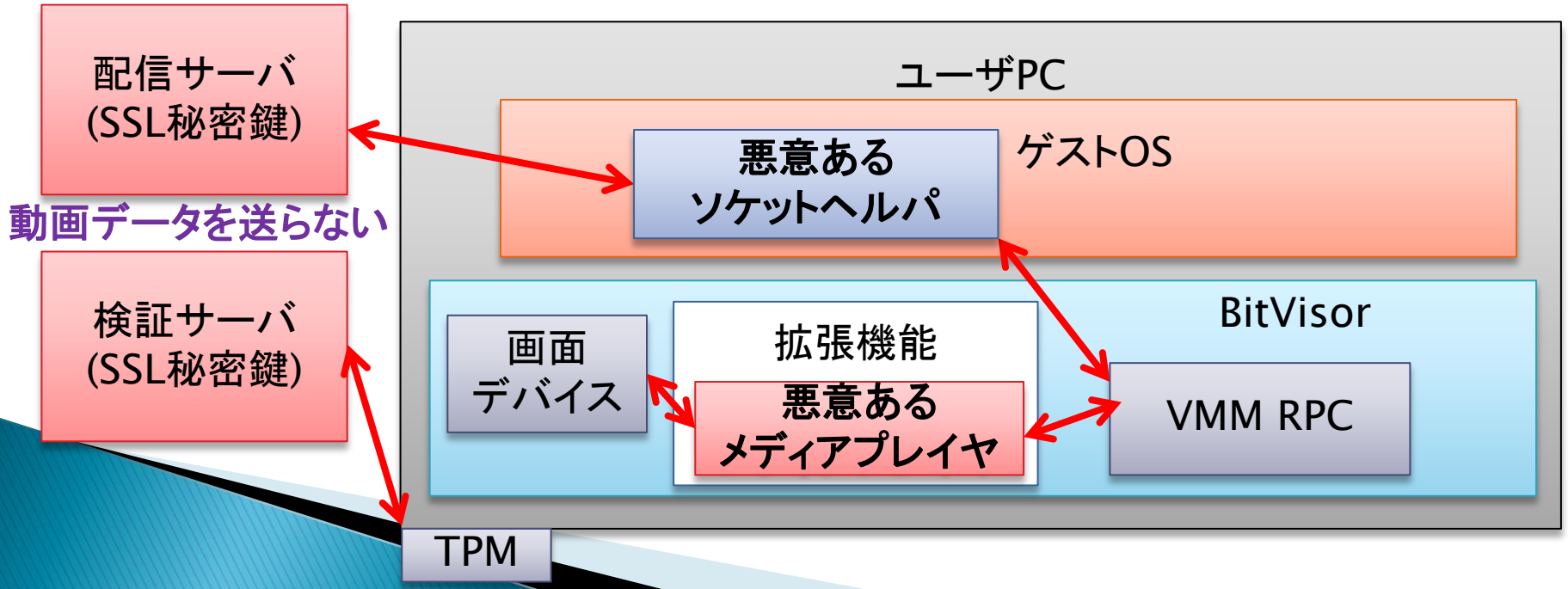
⇨ BitVisorがメモリ保護違反でメディアプレイヤを強制終了



脅威モデル（3/3）

- ユーザがメディアプレイヤとソケットヘルパを改変し、画像データをコピーする攻撃
  - BitVisorの完全性は配信サーバから検証可能

⇨ BitVisorがロードする拡張機能のハッシュ値をTPMへ保存することでメディアプレイヤの完全性も検証可能



本手法と既存手法の比較

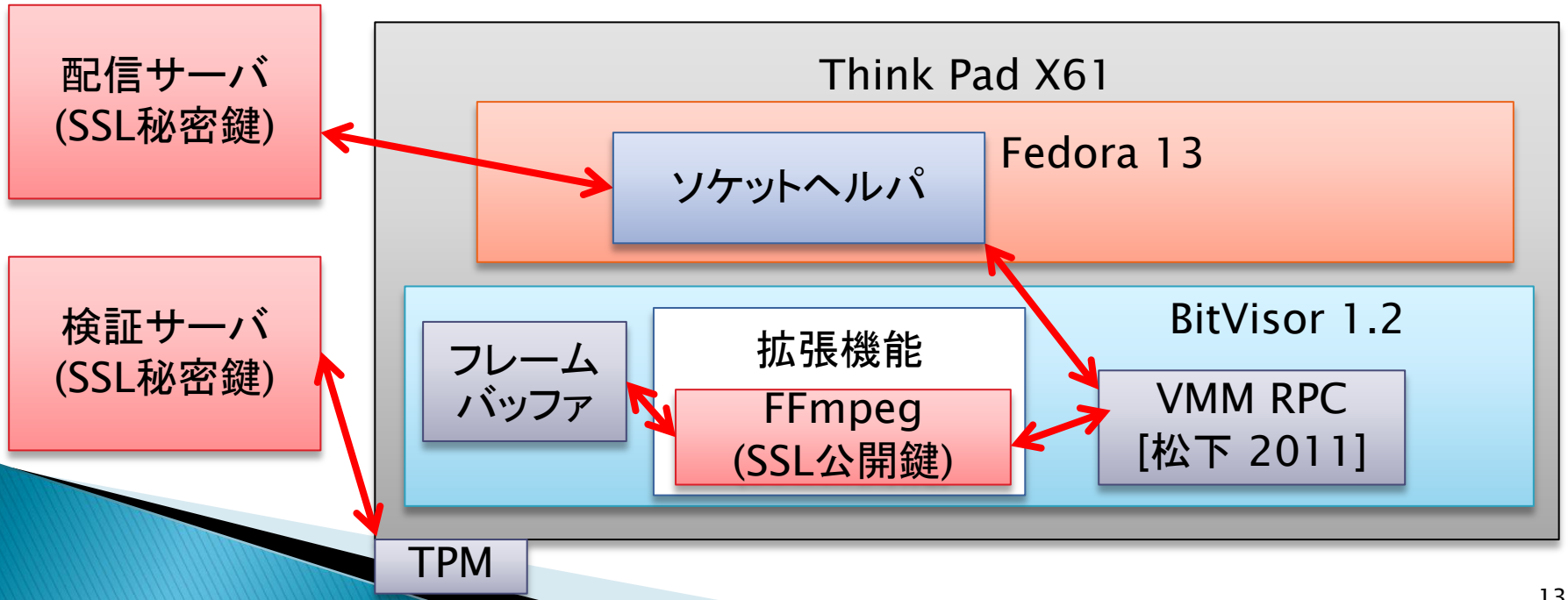
	実行権限	著作権保護
Windows Media DRM	**低い**（ユーザプロセスレベル）	破れる（リバースエンジニアリングの手法）
PlayStation3	高い（VMMレベル）	**破れない**（VMMの権限）
本手法	**低い**（ユーザプロセスレベル、ユーザプロセスより機能が制約された拡張機能を含む）	**破れない**（VMMの完全性を保証し拡張機能として実現）

前提条件：ハードウェアに対する攻撃は想定しない

実行権限が低い＝ユーザは安心して使える

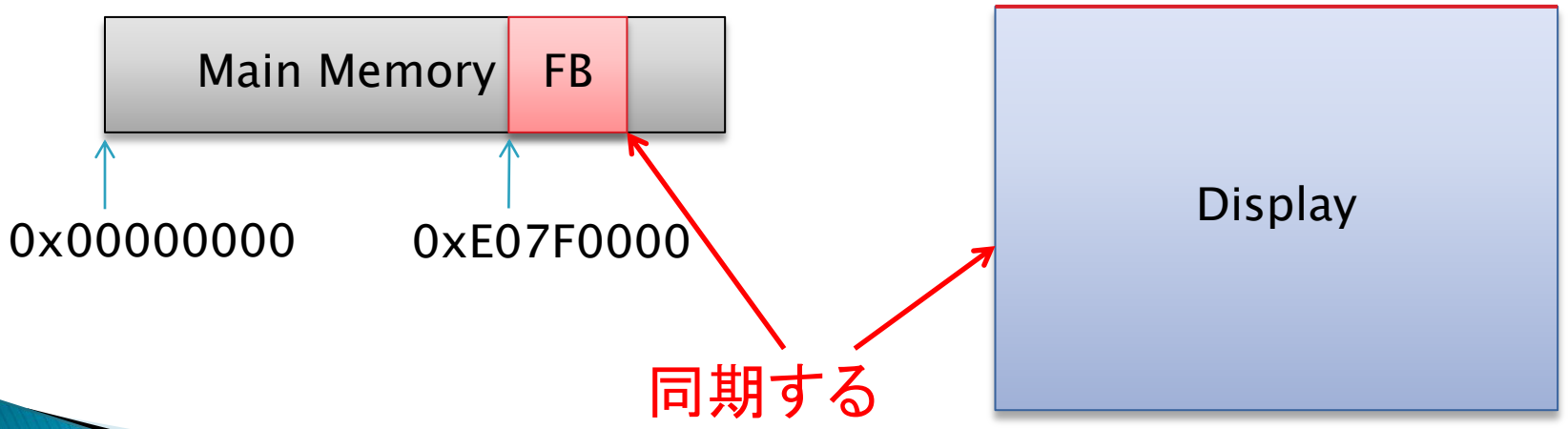
BitVisorによる著作権保護手法の実装

- メディアプレイヤはH.264のデコードができるFFmpegを移植する
- 画面デバイスにはメインメモリ上のフレームバッファを用いる
- ユーザ環境
  - PCにはTPMが搭載されているThink Pad X61
  - ゲストOSにはFedora 13
- BitVisorの完全性を双方から検証可能にする(Trusted Boot)



フレームバッファ(Frame Buffer)とは

- 画面デバイスのうち最も簡単なもの
- ディスプレイに表示される画像データのバッファ
- X61ではメインメモリ上のアドレスにマップされており、そのメモリの内容が一定周期でディスプレイに走査される



著作権保護手法を実現するために必要な実装

1. FFmpegのBitVisorの拡張機能への移植
   - システムコールとライブラリ関数の削除と置き換え
   - Microsoft BMP出力のフレームバッファ出力への改変
2. BitVisorの修正
   - FFmpegによるフレームバッファの専有
   - 浮動小数点演算の実現
3. 拡張機能で実現できる機能の追加
   - コールスタックの保存と復元によりスリープ可能にする
   - ネットワーク通信に利用
4. ソケットヘルパの作成
   - FFmpegと配信サーバ間のSSL通信に利用
   - BitVisorの検証に利用
5. Trusted Bootを用いたBitVisorの完全性の検証
   - 検証サーバを作成しユーザとCP双方から検証可能にする

FFmpegで呼ばれたシステムコール

% time	seconds	usecs / call	calls	errors	syscall
46.43	0.002123	3	646		close
35.02	0.001601	0	5214		write
15.99	0.000731	1	646		open
0.74	0.000034	5	7		munmap
0.72	0.000033	0	648		lseek
0.68	0.000031	0	1582		select
0.28	0.000013	0	165		read
0.13	0.000006	0	1575		gettimeofday
0.00	0.000000	0	6		fstat
0.00	0.000000	0	7		mmap
0.00	0.000000	0	14		brk

（以下略）

サンプル動画をフレームバッファ形式へ変換する際のFFmpegのstrace結果

FFmpegの移植

これらは拡張機能で正しく動かないため、同等の処理をする関数を新たに定義（もしくはBitVisorのものをリンク）

- システムコール一覧
  - write, read, open, close, lseek
  - select, gettimeofday
  - mmap, munmap

- ライブラリ関数一覧
  - memset, strcmp, memcmp, strncmp, strcasecmp
  - toupper,strspn, strcspn, strcpy, sscanf
  - atoi, strtol, snprintf, strstr, snprintf, isdigit
  - memcpy, strtoul, qsort
  - malloc, realloc, free, posix_memalign

拡張機能上での浮動小数点演算の実現(1/2)

- 既存のBitVisorでは浮動小数点演算命令が使えない
  - 浮動小数点演算をした際は例外が発生する
- FFmpegではH.264動画のデコードの際に浮動小数点演算命令を使っている
  - BitVisorの拡張機能上で浮動小数点演算を行いたい

⇒ 拡張機能上のプロセスでのみ浮動小数点演算を可能にする

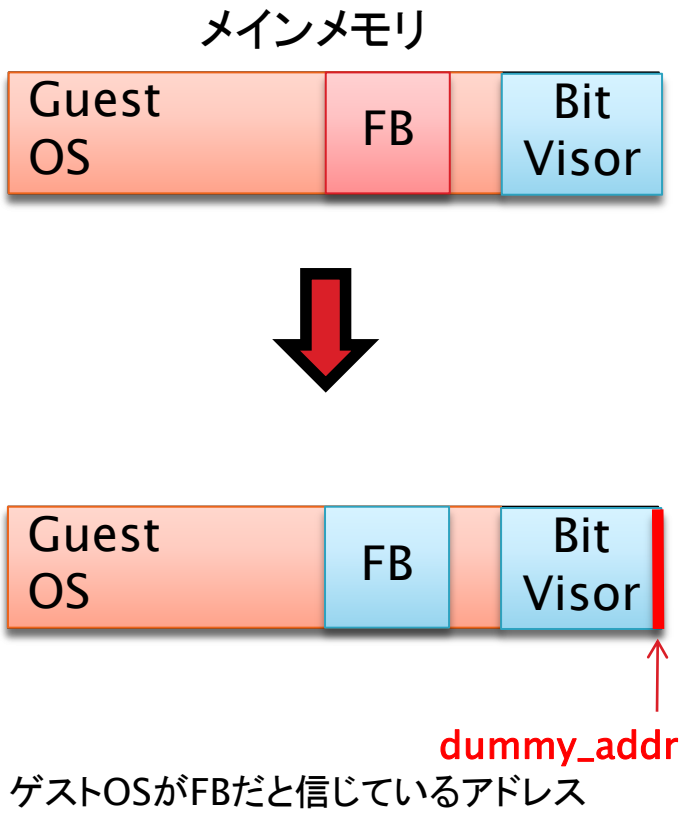
拡張機能上での浮動小数点演算の実現(2/2)

```
1.  static void vmmcall_rpc_stub (void){
2.  /* 省略 */
3.  /* SET CR0_TS_BIT = 0 */;
4.  asm_rdcr0(&cr0);
5.  cr0 &= ~CR0_TS_BIT;
6.  asm_wrcr0(cr0);
7.  save_fpu_registers(env);
8.  ret = vmmcall_rpc(arg.desc, arg.request,
9.  arg.result);
10. /* SET CR0_TS_BIT = 1 */;
11. asm_rdcr0(&cr0);
12. cr0 |= CR0_TS_BIT;
13. asm_wrcr0(cr0);
14. restore_fpu_registers(env);
15. /* 省略 */
16. }
```

A) VMM RPC関数が呼ばれる
B) CR0レジスタのTS_BITをクリア
C) FPU関連のレジスタの保存
D) FFmpegへ処理を移す
E) CR0レジスタのTS_BITをセット
F) FPU関連のレジスタの復元

フレームバッファの専有

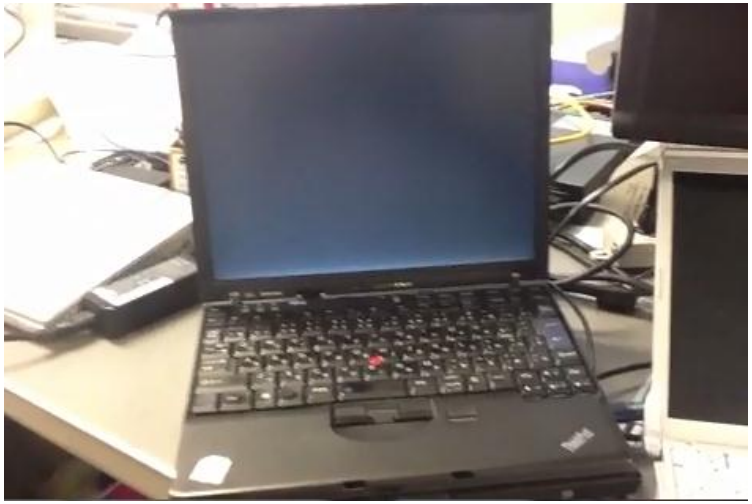
```
1.  u64
2.  gmm_pass_gp2hp (u64 gp, bool *fakerom)
3.  {
4.      bool f;
5.      u64 r;
6.
7.      if (phys_in_vmm (gp)) {
8.              r = phys_blank;
9.              f = true;
10.     } else if (phys_in_fb (gp)) {
11.             r = dummy_addr;
12.             f = false;
13.     } else {
14.             r = gp;
15.             f = false;
16.     }
17.     if (fakerom)
18.             *fakerom = f;
19.     return r;
20. }
```



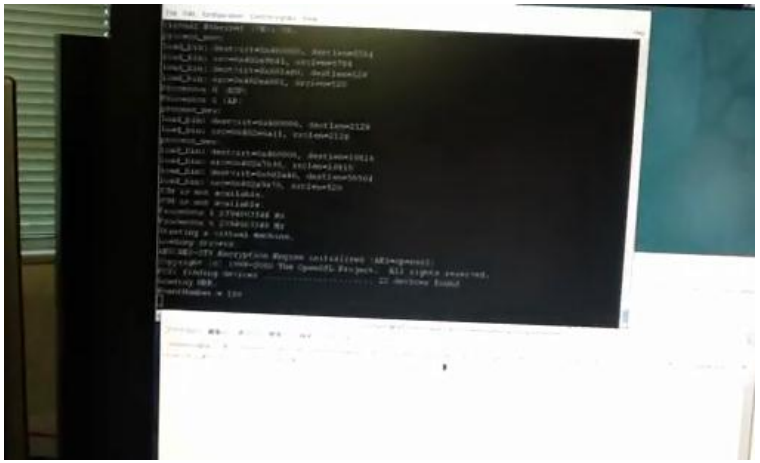
ゲストOSが保持するフレームバッファの物理アドレスを判定し
VMM上の意味のないメモリへのアクセスに変更

デモ

BitVisorの拡張機能上でFFmpegを動かす



BitVisorが入っているノートPC
（ノートPC：X61　ゲストOS：Fedora 13）
フレームバッファにはゲストOSから
アクセスできないため何も表示されない

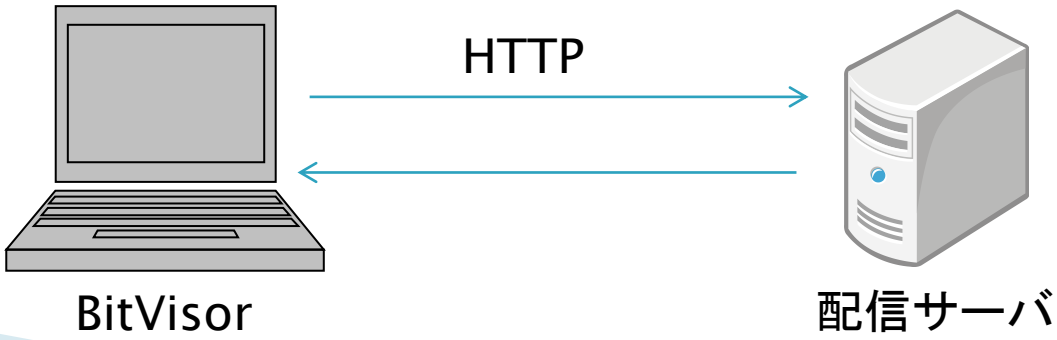


デスクトップPC
sshで左のノートPCへ入り操作している
黒ウィンドウがシリアル経由のBitVisorのログ
白ウィンドウがsshのシェル

拡張機能でのスリープの実現

- レジスタの保存と復元を行う関数を実装
  - extern int env_save (env_t *e);
  - extern void env_restore (env_t *e);

⇒これにより、FFmpegが動画ファイルをreadする際、該当箇所で処理を止め、ソケットヘルパ経由でのHTTPアクセスへ変更することが可能



ソケットヘルパのインターフェース

```
/* request from ffmpeg */
struct rpc_ffmpeg_result {
        u8 req_type ;// 0: nothing , 1: socket
        u8 syscall_num ;// 1: read , 2: write , 3: connect
        u32 xid ;
        u32 sys_args [10];
        char inet_addr [32];
        u16 inet_port ;
        u32 ret ;
        u32 buf_len ;
        u8 *buf ;
};
```

writeの際はbufにHTTPリクエストが格納されている

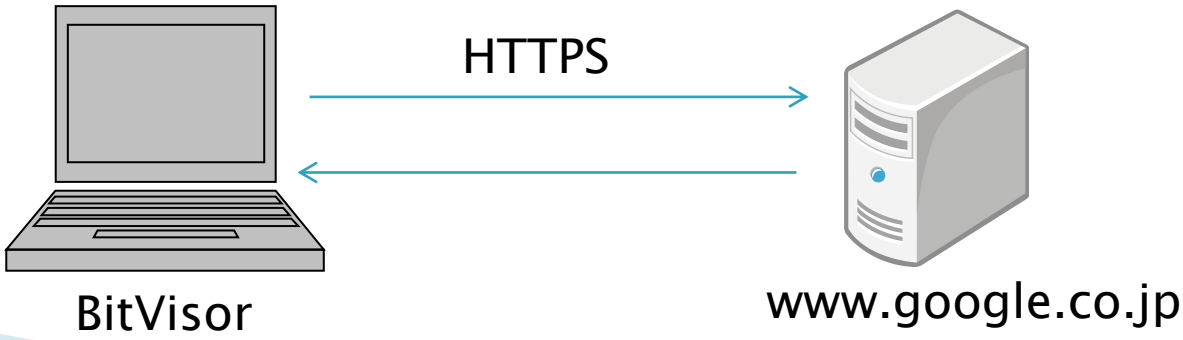
現在、ソケットヘルパを経由したHTTPによる
動画のストリーミング再生に成功している

HTTPSによる通信

- ライブラリの移植
  - OpenSSL (1.0.1c)
    - 現在の実装ではtime(2), rand(2)は常に0を返す
  - ptmalloc2(glibc-2.16.0)
    - sbrk(2)も実装、現在はBSS領域から静的にメモリ確保

⇒これにより、拡張機能からソケットヘルパを介したHTTPS通信が可能となった

(googleのトップページの取得に成功)



関連研究

- ADvisor [小川 2011]
  - グラフィックアクセラレータ内にあるフレームバッファへ画像データを書き込むことで広告表示
  - ゲストOSから広告画像へのアクセスが可能
- Shadowall [尾上 2009]
  - VMで実行されるアプリケーションのデータを保護する研究
- CloudVisor [Zhang 2011]、VMCrypt [田所 2011]
  - VMMでVM間のメモリやディスク内容を保護する研究
- FBCrypt [江川 2011]、FBCrypt-V [西村 2012]
  - リモートと操作者の通信チャネルを保護する研究

本研究では、BitVisorにより画面デバイスを専有した

まとめ

- BitVisorによる動画像を対象とした著作権保護
  - BitVisorの拡張機能でFFmpeg(メディアプレイヤ)を動かした
  - メディアプレイヤは権限の制約された拡張機能として実行されるためユーザは安心して利用できる
  - BitVisorの拡張機能からフレームバッファを専有した
  - ソケットヘルパを介した動画のストリーミング再生に成功した
  - 拡張機能からHTTPSによる通信に成功した
- 今後の課題
  - OpenSSLで用いるrand(2)とtime(2)への対応
  - 完全性が検証されたBitVisor環境への動画像の配信